Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0117 Programación Bajo Plataformas Abiertas

MSc. Marco Villalta Fallas - II Ciclo 2021

Laboratorio #5

C:Funciones, recursión, lectura y escritura de archios

Instrucciones Generales:
Este laboratorio se puede realizar de manera individual o en parejas.
El laboratorio debe de entregarse antes del 2 de noviembre a las 23:59.
Utilice capturas de pantalla para demostrar la funcionalidad.
Entregue un archivo comprimido que incluya un directorio llamado `informe` con el PDF del informe y un directorio llamado `src` con los archivos de código fuente que lleven a la solución. Cualquier otro formato o entrega tardía no se revisará y el laboratorio tendrá una nota de cero.

Documente e investigue los algoritmos solicitados, en el reporte final se debe incluir un diagrama de flujo de la implementación realizada. Cualquier programa que no se compile correctamente no tendrá puntaje. En la calificación de este laboratorio se tomará en cuenta el uso del control de versiones git y la documentación en Doxygen

Realice un programa principal que compruebe la funcionalidad de las siguientes funciones:

1. Capicúa (Si trabaja de forma individual no es necesario realizar este problema)

Realice una función en C que reciba cualquier número entero y determine si es un capicúa o no, en caso de serlo la función debe devolver un 1, caso contrario un 0 (*Sugerencia: Realice otra función que determine la cantidad de dígitos*).

2. Métodos de ordenamiento

Realice una función en C que ordene descendentemente un arreglo de 1500 elementos utilizando el método *bubble sort*. Debe generar de forma aleatorio el arreglo (cuyo rango va de 0 a 100) e imprimir el tiempo que dura en hacer el ordenamiento.

Realice una función en C que ordene descendentemente un arreglo de 1500 elementos utilizando el método *quick sort*. Debe generar de forma aleatorio el arreglo (cuyo rango va de 0 a 100) e imprimir el tiempo que dura en hacer el ordenamiento.

Cuál método es más rápido? (En el reporte debe indicar cuanto tiempo duro cada algoritmo)

3. Aplicación de filtro borroso

Realice un programa que aplique un filtro borroso a un "selfie"suyo. Para esto debe cumplir con lo siguiente:

- Utilice la biblioteca CCV (`https://libccv.org/`)

- Investigue que es una biblioteca estática y una biblioteca dinámica
- Para compilar tanto la biblioteca CCV como el programa utilice el compilador clang
- Para compilar el programa incluya como parámetros al compilador la bandera -I para indicar donde se encuentra el archivo de encabezado de la biblioteca de CCV, la bandera -L para indicar donde se encuentra el archivo de la biblioteca de CCV, incluir la opción para enlazar las biblioteca matemática, PNG, JPEG y CCV. Para la compilación es importante respetar el orden en que se pasan estas opciones al compilador.
- Al programa se le debe pasar como primer parámetro la ubicación de la imagen fuente (selfie) y como segundo parámetro la imagen resultante (borrosa).
- El programa debe abrir la imagen fuente, aplicar un filtro borroso (blur) con un factor de sigma igual a 5.5, finalmente debe guardar la imagen borrosa.

4. Solución de un Laberinto

Se tiene la siguiente representación de un laberinto:

```
####################################
#E#  ##  ######## # #      #     # #
# #   #             # #        #   #
#   # ##### ## ###### # #######  # #
### # ##    ##      # # #     #### #
#   #    #  #######   #   ###    #S#
####################################
```

En ella, el símbolo numeral (`#`) representa las paredes, mientras que la letra `E` y `S` representan entrada y salida respectivamente. Los espacios en blanco muestran los corredores que se pueden recorrer.

Para resolver el ejercicio, debe seguir los siguiente pasos:

1. Leer el laberinto desde un archivo de texto plano cuya ubicación se pasa como primer argumento al programa, convertirlo a una representación con arreglo(s) en donde cada cuadrícula del laberinto representa una entrada de la lista bidimensional.
   - Si la cuadrícula es un corredor por el cual se puede transitar, su representación en la lista bidimensional debe ser un 0.
   - Si la cuadrícula es una pared, su representación en la lista bidimensional debe ser un 1.
   - Si la cuadrícula es la salida, su representación en la lista bidimensional debe ser un 2.
   - Si la cuadrícula ya ha sido visitada por el algoritmo, su representación en la lista bidimensional debe ser un 3.
2. Implemente una lógica con una función recursiva que sea capaz de recorrer el laberinto y encontrar una solución.
3. Finalmente, una ves solucionado el laberinto, convierta la representación de la lista bidimensional de nuevo a la forma original y escriba el contenido en un archivo de texto plano cuya ubicación se pasa como segundo argumento al programa.

   La solución debe de verse similar a la siguiente forma:

```
####################################
#S#  ##  ######## # #oooooo#  ooo# #
#o#ooo#    oooo     #o#   ooooo#ooo#
#ooo#o#####o##o######o# #######  #o#
### #o##oooo##oooooo#o# #     ####o#
#   #oooo#  ######ooo#   ###    #E#
####################################
```